1) Python Programming

```
def print_every_other_word(filename):
   with open(filename) as f:
       lines = f.readlines()
       for line in lines:
           words = line.split()
           counter = 0
           for word in words:
               if counter % 2 == 0:
                   print(word, end=" ")
               counter += 1
           print()
```

2.a.i) `max_atk_difference_manual`

```
def max_atk_difference_manual(data):
    if len(data) < 2:
        return None
    smallest = data[0]['atk']
    largest = data[0]['atk']
    for row in data:
        atk = row['atk']
        if atk < smallest:
            smallest = atk
        elif atk > largest:
            largest = atk
    return largest - smallest
```

2.a.ii) `max_atk_difference_pandas`

```
def max_atk_difference_pandas(data):
    if len(data) < 2:
        return None
    return data['atk'].max() - data['atk'].min()
```

2.b) `best_matchup_manual`

```
def best_matchup_manual(data, index):
    pokemon = data[index]
    result = {}
    for row in data:
        t = row['type']
        n = row['name']
        if t != pokemon['weakness'] and n!= pokemon['name']:
            if t in result:
                if row['atk'] < result[t]['atk']:
                    result[t] = row
            else:
                result[t] = row

    return {r['type']: r['atk'] for r in result}
```

3.a) Train a classifier

```
dogs = pd.read_csv('dogs.csv')
X = dogs.loc[:, (dogs.columns != 'type') & (dogs.columns != 'breed')]
y = dogs['type']
model = DecisionTreeClassifier()
model.fit(X, y)
```

3.b) Featuring

The breed does not seem like an informative feature since there are too many distinct categorical values. Since we don't have a large dataset, it might cause the model to overfit since it only has one example of each time.

3.c) Making Predictions

i. doggo
ii. pupper

4.a) Implementing `Rectangle`

```
class Rectangle:
     def __init__(self, w, l):
          self._w = w
          self._l = l

     def area(self):
          return self._w * self._l

     def scale(self, scale_factor):
          self._w *= scale_factor
          self._l *= scale_factor

     def __eq__(self, other):
          return self._w == other._w and self._l == other._l
```

4.b) Using `Rectangle`

```
r1 = Rectangle(2, 3)
r1.scale(2)
print(r1.area())
r2 = Rectangle(4, 6)
if (r1 == r2):
     print("Same")
else:
     print("Different")
```

5) Efficiency

a. O(n)
b. O(n)
c. O(1)
d. O(n^2)